

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-280989

(43)Date of publication of application : 03.10.2003

(51)Int.Cl.

G06F 12/14
G06F 1/00
G06F 9/46
G06F 12/10
G06F 15/78

(21)Application number : 2002-079155

(71)Applicant : TOSHIBA CORP

(22)Date of filing : 20.03.2002

(72)Inventor : HASHIMOTO MIKIO

YAMAGUCHI KENSAKU

SHIRAKAWA KENJI

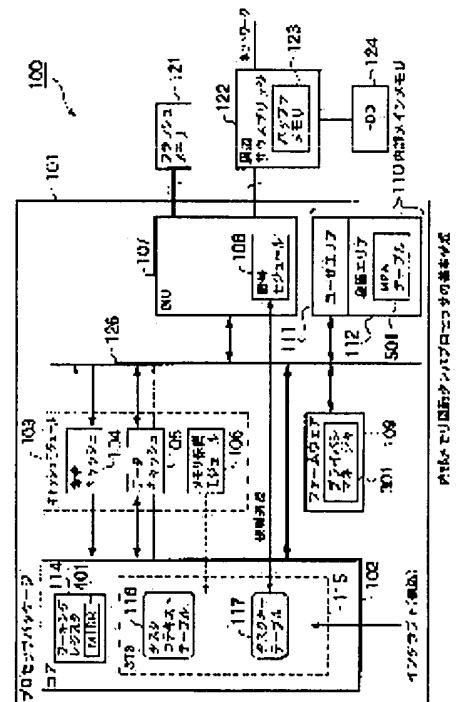
FUJIMOTO KENSAKU

(54) INTERNAL MEMORY TYPE TAMPER-RESISTANT PROCESSOR AND SECRECY PROTECTION METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To realize an efficient internal data management able to prevent an unauthorized alteration under a multitask environment.

SOLUTION: This processor is provided with a decryption means to decrypt a plurality of programs encrypted with a different encryption key, an internal memory 110 to store the decrypted programs in units of memory page respectively, a requested secrecy protection attribute retention part which configures a requested secrecy protection attribute requested for every memory page as an access target exclusively from other tasks to store when reading the program into other memory page of the internal memory and executing it as a task, a memory secrecy protection attribute retention part which stores a memory secrecy protection attribute for each memory page, a secrecy protection management part which configures the memory secrecy protection attribute onto the memory secrecy protection attribute retention part for the memory page as an access target when



内部メモリ型耐改竄プロセッサの構成図

executing the task, and a memory protection means to permit the access when the requested secrecy protection attribute stored in the requested secrecy protection attribute retention part and the memory secrecy protection attribute stored in the memory secrecy protection attribute retention part are compared and matched.

LEGAL STATUS

[Date of request for examination] 19.05.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

【特許請求の範囲】

【請求項 1】 異なる暗号鍵で暗号化された複数のプログラムを各々復号する復号化手段と、
前記復号化手段により復号された複数のプログラムを、
メモリページ単位で各々平文にて格納するための内部メモリと、

前記プログラムを前記内部メモリの他のメモリページに読み込み、タスクとして実行する際に、該タスクがアクセス対象の前記メモリページ毎に要求する要求秘密保護属性を、他のタスクとは排他的に設定し格納する要求秘密保護属性保持部と、

前記メモリページ毎に、当該メモリページのメモリ秘密保護属性を格納するメモリ秘密保護属性保持部と、

前記タスク実行時に、当該タスクがアクセス対象の前記メモリページについて、前記メモリ秘密保護属性保持部に、前記メモリ秘密保護属性を設定する秘密保護管理部と、

前記要求秘密属性保持部に格納された前記要求秘密保護属性と、前記メモリ秘密保護属性保持部に格納された前記メモリ秘密保護属性とを比較し、一致した場合に当該メモリページへのアクセスを許可するメモリ保護手段とを具備したことを特徴とする内部メモリ型の耐タンパマイクロプロセッサ。

【請求項 2】 前記メモリ秘密保護属性保護部と、前記要求秘密保護属性保護部は、前記タスクを識別するためのタスク識別子を更に格納し、

前記メモリ保護手段は、前記要求秘密保護属性と前記メモリ秘密保護属性を比較するとともに、各々対応する前記タスク識別子を比較し、前記要求秘密保護属性と前記メモリ秘密保護属性が一致し、かつ各々対応する前記タスク識別子が一致した場合にのみ当該メモリページへのアクセスを許可することを特徴とする請求項 1 に記載の内部メモリ型の耐タンパマイクロプロセッサ。

【請求項 3】 前記秘密保護管理部は、前記タスクが発行する命令に基づいて、前記メモリ秘密保護属性保持部に前記メモリ秘密保護属性を設定することを特徴とする請求項 1 に記載の内部メモリ型の耐タンパマイクロプロセッサ。

【請求項 4】 前記秘密保護管理部は、前記タスクが前記アクセス対象であるメモリページにアクセスする際に、前記比較結果に基づいて前記メモリ秘密保護属性を設定することを特徴とする請求項 1 に記載の内部メモリ型の耐タンパマイクロプロセッサ。

【請求項 5】 前記秘密保護管理部は、前記タスクがアクセス対象の前記メモリページ毎に要求する要求秘密保護属性が、秘密保護を要求し、該アクセス対象のメモリページのメモリ秘密保護属性が非秘密保護の属性であった場合には、前記メモリ秘密保護属性を秘密保護の属性とし、当該メモリページを初期化することを特徴とする請求項 4 に記載の内部メモリ型の耐タンパマイクロプロ

セッサ。

【請求項 6】 前記内部メモリの仮想アドレスを物理アドレスに変換するアドレス変換手段を更に具備し、
前記メモリ秘密保護属性保持部は、前記メモリページの物理アドレス毎に前記メモリ秘密保護属性を格納し、
前記メモリ秘密保護属性保持部と、前記要求秘密保護属性保持部は、それぞれアクセス対象であるメモリページの仮想アドレスを更に格納することを特徴とする請求項 1 に記載の内部メモリ型の耐タンパマイクロプロセッサ。

【請求項 7】 前記メモリ保護手段は、前記仮想アドレスから前記物理アドレスへの変換時に、前記メモリ秘密保護属性保持部の仮想アドレスと前記要求秘密保護属性保持部の仮想アドレスとを比較し、一致した場合にアドレス変換を行うことを特徴とする請求項 6 に記載の内部メモリ型の耐タンパマイクロプロセッサ。

【請求項 8】 前記内部メモリは複数のタスクによって共有される共有領域を有し、

前記メモリ秘密保護属性と、前記要求秘密保護属性は、前記共有のための秘密値を各々含み、

前記メモリ保護手段は、前記メモリ秘密保護属性に含まれる前記秘密値と、前記要求秘密保護属性に含まれる前記秘密値とを比較し、一致した場合にのみ前記共有領域へのアクセスを許可することを特徴とする請求項 1 に記載の内部メモリ型の耐タンパマイクロプロセッサ。

【請求項 9】 異なる暗号鍵で暗号化された複数のプログラムをマイクロプロセッサ内部に読み込み各々復号し、

復号された複数のプログラムを、メモリページ単位で各々平文で内部メモリに格納し、

このプログラムを前記内部メモリの他のメモリページに読み込み、タスクとして実行する際に、該タスクがアクセス対象の前記メモリページ毎に要求する要求秘密保護属性を他のタスクとは排他的に設定し、

前記メモリページ毎に、当該メモリページのメモリ秘密保護属性を設定し、

前記要求秘密保護属性と前記メモリ秘密保護属性とを比較し、一致した場合に当該メモリページへのアクセスを許可することを特徴とする内部メモリの秘密保護方法。

【請求項 10】 異なる暗号鍵で暗号化された複数のプログラムをマイクロプロセッサ内部に読み込み各々復号し、

復号された複数のプログラムを、メモリページ単位で各々平文で内部メモリに格納し、

このプログラムを前記内部メモリの他のメモリページに読み込み、タスクとして実行する際に、該タスクがアクセス対象の前記メモリページ毎に要求する要求秘密保護属性と当該タスクを識別するタスク識別子を関連付けて、かつ他のタスクとは排他的に設定し、

前記メモリページ毎に、当該メモリページのメモリ秘密

10

20

30

40

50

保護属性を設定し、前記要求秘密保護属性と前記メモリ秘密保護属性が一致し、かつ前記メモリページのタスク識別子と前記実行されるタスクのタスク識別子が一致した場合にのみ当該メモリページへのアクセスを許可することを特徴とする内部メモリの秘密保護方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、パッケージ内部に、複数のプログラムとその作業領域を格納可能な大容量の内部メモリを有し、マルチタスクのプログラム実行環境下で、実行コードや処理対象データの不正な改変を防止することのできる耐タンパマイクロプロセッサに関する。

【0002】

【発明の背景】情報システムネットワークとして、オープンシステムが広く普及している。オープンシステムでは、PCなどの一般ユーザ向けコンピュータのハードウェアやシステムプログラム（あるいはOS）の情報が開示されており、エンドユーザは、システムプログラムを変更して望んだ改良を加えることができる。このような環境で、アプリケーションプログラムが扱う情報の著作権や、プログラム自体の著作権を保護するためには、システムのOSがアプリケーションに対して敵対的動作をとり得ることを前提として、プログラムの秘密を守るハードウェアが必要となってくる。こうしたハードウェアを、特にマイクロプロセッサとして実現する試みが提案されている（特願平2000-35898号、Lie et al., "Architectural Support for Copy and Tamper Resistant Software", ; Computer Architecture News 28 (5):p168-等）。

【0003】これらのマイクロプロセッサは、マルチタスク環境で、プログラムとそれが扱う情報の覗き見や改変を防止するために、情報を暗号化する機能を備えている。このようなマイクロプロセッサを耐タンパマイクロプロセッサと呼ぶ。

【0004】耐タンパマイクロプロセッサの主な目的は、エンドユーザのシステムで動作するアプリケーションの保護を通じて、プログラムやコンテンツ、ネットワークサービスの権利者の権利を保護することである。より具体的には次の3つが挙げられる。

【0005】（1）プログラムに実装されたアルゴリズムの秘密保護

（2）プログラムに埋め込まれたトレードシークレットおよびコンテンツの、秘密または不正複製からの保護

（3）プログラム動作、処理結果の改変からの保護
プログラムに実装されたアルゴリズムの保護は、プログラムの権利者の保護に必要であり、プログラムに埋め込まれたトレードシークレットの保護は、プログラムが扱い著作権保護されたコンテンツの不正コピーを防止する

のに必要である。ネットワークサービスを利用するプログラムが、課金情報をサーバとやりとりするようなアプリケーションでは、課金情報の送信動作を正しく実行し、不正な改変を防止することが、サービスプロバイダの権利の保護に重要である。実際の例として、パーソナルコンピュータ（PC）でDVDを再生するプログラムが解析され、そこからDVDの暗号を解読するためのトレードシークレットが読みとられて不正にDVDをコピーするプログラム（DeCSS）が作られたことはよく知られている。

【0006】ここで、現在提案されているオープンシステムでのアプリケーションプログラムの秘密保護機構について、本願発明者らの提案（特願2000-35898号、特願2000-135010号、特願2000-333635号、特願2001-024480号）に基づき、簡単に説明する。紹介する例では、ひとつのシステム上で疑似並列的に複数のプログラムベンダのプログラムが実行され、それぞれがそのシステム資源を管理するOSからも独立に秘密を保持できることが特徴である。このような環境を「マルチパーティアプリケーション保護環境」と呼ぶこととする。

【0007】図11は、これまでに提案されているマルチパーティのアプリケーション保護環境の概要を示す。ターゲットシステム（たとえばPC）1102は、耐タンパのマイクロプロセッサ1103を内蔵し、ハードディスクのような2次記憶1107と、プロセッサ外部のメモリ1108を有する。システム1102と、ユーザ1112で、システム環境1101を構成する。

【0008】複数の異なるプログラムベンダ1121-1~1121-nは、暗号化したプログラム1123-1~1123-nを、それぞれネットワークを介してターゲットシステム1102に配信する。暗号化されたプログラムを保護プログラムと呼ぶ。

【0009】ベンダから配信された暗号化プログラムは、ターゲットシステム1102の2次記憶1107に蓄積され、実行時に外部メモリ1108上に確保された領域1109に読み込まれる。外部メモリ上では、プログラムは未だ暗号化された状態である。暗号化プログラムは、外部メモリ1108からマイクロプロセッサ1103に取り込まれてはじめて復号される。復号処理は、各プログラムに対応してベンダから送られてきた鍵を用いて、保護機能1106で行なわれる。復号されたプログラムはキャッシュメモリ1104に読み込まれる。プログラムの復号とキャッシュメモリ1104への読み込みは、一度に全体が読み込まれるわけではなく、プログラムの実行にしたがって部分的に行われる。キャッシュメモリに読み込まれたプログラム部分は平文状態である。

【0010】マイクロプロセッサ1102の内部では、保護機能1106によってプログラムは平文で取り扱わ

10

20

30

40

50

れ、OS 1110が介在する余地はない。また、キャッシュメモリ 1104の内容は、マイクロプロセッサ 1103の仕様に定められた操作を除いて外部から直接読み出すことはできない。

【0011】一方、近年、集積回路技術の発達により、マイクロプロセッサと同一パッケージ内に大容量のメモリを搭載することが可能になっている。このような内部メモリ型のプロセッサでは、最大メモリ用量の制限はあるものの、従来の外部メモリを前提とした耐タンパプロセッサでオーバヘッドとなっていたメモリ読み込み、書き出し時の暗号処理負荷を低減できる可能性がある。

【0012】しかし、メモリが内部メモリとしてマイクロプロセッサ内部に配置されたとしても、メモリの資源管理はあくまでもOSの管理下にあるので、OSによる敵対動作が十分に考えられる。したがって、内部メモリ型のマイクロプロセッサでは、OSの管理下でのプログラムの秘密保護が必要となってくるが、具体的な秘密保護の実現手段は何ら提案されていない。

【0013】さらに、内部メモリ型のマイクロプロセッサで、単一のプログラムのみならず、複数の異なる暗号プログラムを擬似並列的に処理する場合に、複数プログラムの秘密保護を個別に保証する必要があるが、これについても、なんら提案されていない。

【0014】

【発明が解決しようとする課題】内部メモリ型の耐タンパマイクロプロセッサで、秘密保護を保証しつつマルチタスク処理を実現するためには、

(1) 複数プログラムを意図的改変から保護できるメモリ操作の統合性(integrity)

(2) OSによる資源管理と内部メモリの秘密保護の両立

(3) タスクとOSとの通信を考慮する必要がある。

【0015】具体的には、あるタスクが内部メモリのあるアドレスに対してn回のメモリ操作をおこなう場合、その過程で別のタスクによるメモリ操作からも、OSによる意図的な攻撃からも保護されなければならない。タスクの実行は数多くの命令のステップからなり、それらが正しく実行されてはじめて正しい結果が得られるからである。OSを含む攻撃者が、プログラムのある一部分だけを実行する、あるいは一連のメモリ操作のうち特定操作だけを除外する場合、プログラムの動作が恣意的に変更され、正しいプログラム実行ができなくなる。

【0016】また、OSが資源管理の一環として行うメモリページの操作時に、メモリの解放を利用した攻撃が考えられ得るが、このような攻撃を排除すべく、資源管理とプログラムの秘密保護を両立させなければならない。

【0017】さらに、タスクはシステムコールの発する場合、OSとデータ交換するが、このときOSとメモリ

を共有して通信する必要がある。

【0018】ここで、従来の外部メモリ型耐タンパマイクロプロセッサの保護機構を、そのまま内部メモリ型に適用したプロセッサを考えてみる。まずメインの内部メモリは、ブロックまたはページと呼ばれる単位(たとえば4Kバイトページ)に分割される。プロセッサ内部には、メモリページ毎にその秘密保護属性を保持するテーブルが設けられ、テーブルエントリが、対応するメモリページの秘密保護属性を保持する。タスクT。が操作したメモリページの秘密保護属性記述子には、自動的にタスクT。のIDが設定され、OSを含む他のタスクは一切その内容を平文で読み出し、書き換えできなくなる。ただし、OSはメモリページの再利用のために、あるメモリページの秘密保護属性を強制的に解除することができる。メモリページの秘密保護属性が解除されると、ページの内容はすべて消去されたうえで、所定の値に初期化され、もとタスクの秘密は守られる。

【0019】いま、タスクT。が、あるアドレスA。のメモリページM。に対して、M。の初期状態からOp₁~Op_n。までn回のメモリ操作をおこなうとする。初期状態は不定であってよいが、Op₁~Op_n。の操作を確実にこなって、最終状態の結果を得ることが目的である。当然その過程では別のタスクによるメモリ操作が加わってはならない。

【0020】一般タスクによるメモリ操作は、従来の外部メモリ型の保護機構により排除できる。しかし、OSがページM。の秘密保護属性を解除した後に、当該ページを攻撃者の意図するデータに書き換える攻撃は、外部メモリ型の保護機構をそのまま内部メモリ型に適用しても、排除できないし、検出もできない。秘密保護属性解除以前のデータの秘密は保護され得るが、他のデータへの置き換えは防止できないのである。

【0021】結果として、OS特権を持つ攻撃者が、タスクの秘密メモリ内容を意図したように自由に書き換えられることになってしまう。

【0022】メモリページの強制的な秘密保護属性の解除は、タスクの秘密保護にとって脅威となる。しかし、マルチタスクのシステム下では、このような強制解除の機能を排除してタスク自身だけが当該保護属性を解除できる構成とすることはできない。なぜなら、一部のタスクによって、過大なメモリが占有されてしまった場合に、OSがそれを制御できなくなるからである。

【0023】また、OSとタスクが通信するための非保護状態のメモリアクセスと、保護されたメモリアクセスは、共存できるものでなければならない。

【0024】そこで、本願ではこれらの点に留意しつつ、パッケージ内に比較的大容量の内部メモリを持つマイクロプロセッサにおいて、マルチタスクのプログラム実行環境下で、実行コードや処理対象であるデータの不正な改変を防止できる効率的な内部データ管理を実現す

る。

【0025】

【課題を解決するための手段】内部メモリ型耐タンパプロセッサは、以下を備える。

【0026】(a)異なる暗号鍵で暗号化された複数のプログラムを、各々復号する復号化手段；

(b)復号化手段により復号された複数のプログラムを、メモリページ単位で各々平文にて格納するための内部メモリ；

(c)前記プログラムを内部メモリの他のメモリページに読み込み、タスクとして実行する際に、このタスクがアクセス対象のメモリページ毎に要求する要求秘密保護属性を他のタスクとは排他的に設定し格納する要求秘密保護属性保持部；

(d)メモリページ毎に、当該メモリページのメモリ秘密保護属性を格納するメモリ秘密保護属性保持部；

(e)タスク実行時に、当該タスクがアクセス対象のメモリページについて、メモリ秘密保護属性保持部にメモリ秘密保護属性を設定する秘密保護管理部；および

(f)要求秘密属性保持部に格納された要求秘密保護属性と、メモリ秘密保護属性保持部に格納されたメモリ秘密保護属性とを比較し、一致した場合に当該メモリページへのアクセスを許可するメモリ保護手段。

【0027】このような内部メモリ型マイクロプロセッサでは、プログラムやデータが平文状態で内部メモリに格納されるので、メモリの読み出し、書き出しに暗号処理を必要としない分、処理能力のオーバーヘッドを解消できる。

【0028】また複数プログラムを擬似並列的に走らせるマルチタスク環境下で、メモリ資源管理を行うOSが、メモリページの解放を利用して不正処理を意図したとしても、成功しない。タスクがアクセス対象のメモリページに要求する秘密保護属性と、このタスク処理に必要なメモリ領域に設定されている秘密保護属性とが一致しない限り、そのメモリページへのアクセスが許可されないからである。

【0029】このような構成により、複数プログラムの個々の秘密が保護され、OSによる資源管理と各プログラムの秘密保護が両立する。また、メモリページの秘密保護属性を管理することによって非保護領域を確保し、タスクとOSの通信を保証できる。

【0030】

【発明の実施の形態】以下の説明において、マイクロプロセッサ外部のメモリに暗号化された状態で格納されたプログラムを読み込んで処理する従来型の耐タンパプロセッサを「外部メモリ型耐タンパプロセッサ」、マイクロプロセッサのパッケージ内にプログラムを平文状態で格納する耐タンパプロセッサを「内部メモリ型の耐タンパプロセッサ」と称する。

【0031】また本明細書において、プログラムの秘密

保護とは、固有のプログラム鍵で暗号化されているプログラムの作業データが、他のプログラムから読み取られないことはもちろんのこと、攻撃者の意図にそったデータの改変や、プログラムの動作の改変といった攻撃も防止することを意味する。

【0032】＜第1実施形態＞図1は、本発明の第1実施形態に係る内部メモリ型の耐タンパプロセッサの概略構成図である。プロセッサ100は、プロセッサパッケージ101内に、プロセッサコア102と、キャッシュモジュール103と、バスインターフェイスユニット(BIU)107と、プライバシマネージャ(秘密保護管理部)301を含むファームウェア(ROM)109と、内部メモリ110を有し、これらは内部バス126を介して相互接続されている。

【0033】プロセッサコア102は、実行中のプログラムが使用するワーキングレジスタ114と、タスク切り替え部(STS: secure task switcher)115を有する。ワーキングレジスタ114は、MTR(memory type range)レジスタ401を含む。MTRレジスタ401は、タスクがアクセス対象である内部メモリ領域(すなわちメモリページ)に秘密保護を要求するかどうかを指定する要求秘密保護属性(RPA: requested protection attribute)を格納する。タスク切り替え部115は、タスク対応に実行中のレジスタ状態を保持するコンテキストテーブル116と、タスク対応にプログラム暗号鍵などの鍵情報を保持する鍵テーブル(タスクキーテーブル)117を有する。

【0034】ファームウェア109には、図示はしないが、プロセッサ100に固有の秘密鍵が格納されている。ファームウェア109は、プライバシマネージャと呼ばれる秘密保護管理部301を含み、内部メモリ110の各メモリページのメモリ秘密保護属性(MPA: memory protection attribute)を管理する。プライバシマネージャ301は、たとえばファームウェアに格納されるプログラムとして実現される。

【0035】内部メモリ110は、大きく2つの領域に分割される。ユーザエリア111と秘匿エリア112である。ユーザエリア111には、OSやそのシステムエリアを含むユーザプログラムが、実行時に格納される。秘匿エリア112は、プライバシマネージャ301のみが操作できる領域で、メモリ秘密保護属性(MPA: memory protection attribute)テーブル501を格納する。MPAテーブル501は、メモリページごとにその秘密保護属性を格納する。上述したプライバシマネージャ301は、タスクが発行する秘密保護命令に応じて、対応するメモリページに関し、MPAテーブル501にメモリ秘密保護属性を設定する。秘匿エリア112にはOSによる操作がいつさい及ばない。

【0036】キャッシュモジュール103は、命令キャッシュ104と、データキャッシュ105と、メモリ保

護モジュール（メモリ保護手段）106を有する。データキャッシュ105は、プログラムの実行時（すなわち対象領域へのアクセス時）に、内部メモリ110のユーザエリア112のページ内容と、それに対応してMPAテーブル501に保持された秘密保護属性とを読み込む。メモリ保護モジュール106は、データキャッシュ105のキャッシュタグに書き込まれたメモリ秘密保護属性（MPA）と、コア102のMTRレジスタ401から出力されたタスク指定の要求秘密保護属性（RPA）とを比較し、これら2つの属性が一致したときのみ、対象となるメモリページへのアクセスを許可する。

【0037】BIU107は、暗号モジュール（復号化手段）108を有し、外部との入出力データを暗号化する。プロセッサ100は、BIU107を介して外部と入出力を行う。外部バスでプロセッサと接続された周辺機器ブリッジチップ122は、内部にバッファメモリ123を有する。周辺機器ブリッジチップ122はフラッシュメモリ121およびHDD125とそれぞれ接続されている。

【0038】図2は、内部メモリマップを示す。ユーザエリア111と、プライバシマネージャのための領域301は、メモリマップ上で隔離される。メモリマップの前半部分はユーザエリアに割り当てられ、後半部分はプライバシマネージャの領域に割り当てられる。ユーザエリア111におかれた外部プログラムは、プライバシマネージャ301を含むファームウェア領域109に対してアクセスすることはできない。

【0039】第1実施形態では、説明の単純化のため、仮想アドレス機構をもたない実アドレス方式のプロセッサを仮定する。このような耐タンパプロセッサの動作を以下で説明する。

【0040】プログラムの読み込みとタスクの生成

図3は、図1のマイクロプロセッサ100によるプログラムAの読み込みとタスク生成の手順を示す概略図である。プログラムAは、プロセッサ100によって実行される前は、暗号化された状態でマイクロプロセッサ外部のフラッシュメモリ121（図1）やHDD124（図1）に格納されている。

【0041】プログラムAの読み込み以前に、OSや別のプログラムが読み込まれて生成されたタスク1が、ユーザエリア111に読み込まれている。OSはユーザエリア111のアドレス0、Pから始まる領域に、タスク1はアドレスXから始まる領域に読み込まれている。

【0042】プログラムAは配布鍵302とプログラム本体303で構成される。プログラムAの読み込みにあたって、OSは所定の手順により、暗号化された状態のプログラムAを周辺機器ブリッジチップ122のバッファメモリ123に読み込む。次にOSは、バッファメモリ123のプログラムのアドレス、内部メモリにおけるプログラムの配置アドレスY、およびタスクID（タス

クk）を指定して、タスク生成命令を発行する。

【0043】タスク生成命令が発行されると、制御はOSからプライバシマネージャ301に移る。プライバシマネージャ301は、たとえばファームウェア109に格納されたプログラムとして実装され、プログラムの秘密を守るために外部プログラムであるOSよりも強い特権を持つ。プライバシマネージャ301だけが、内部メモリの秘匿エリア112やコンテキストテーブル116、タスクキーテーブル117を操作できる。

【0044】プライバシマネージャ301は、バッファメモリ123からプログラムAの配布鍵302を取得し、ファームウェア109に格納されたプロセッサ秘密鍵（不図示）を使って、暗号モジュール108を介して配布鍵302を復号し、プログラム鍵を取り出す。このプログラム鍵を、タスクキーテーブル117のタスクID（すなわちタスクk）に対応するエントリ117-kに格納する。次にプログラム鍵によってバッファメモリ123メモリのプログラム本体303を、暗号モジュール108を介して復号し、内部メモリのプログラム配置アドレスYから始まる領域に復号したプログラムを配置する。

【0045】さらに、秘匿エリア112のメモリ秘密保護属性（MPA）テーブル501（図1）のプログラム配置領域に対応するエントリに、タスクID（k）を書き込み、これらの領域に保護属性を与えて、他のタスクからのアクセスを防止する。MPAテーブル501のエントリは、後述するように、メモリのページごとに（たとえば4Kバイトを1ページとして）設定される。それまでに実行されていたタスク1のエントリには、タスクID=1が書き込まれており、今回のプログラムAの読み込みによって、タスクID=kが書き込まれる。

【0046】これでタスク実行の準備が完了し、一度制御がOSに戻された後、OSがタスクID=kの実行開始を指示する命令を発行する。すると、所定のアドレスからプログラムAすなわちタスクkの実行が開始される。なお、第1実施形態ではプライバシマネージャ301はファームウェア109によって実現されるものとしたが、ハードウェアによって実装されてもよい。ハードウェアによる実装の場合は、プライバシマネージャ301のタスク実行準備がなされている間、OSは他のタスクを並行して処理することができ、システムの応答性能が向上する。

【0047】メモリの確保

さて、生成されたタスクkを実行するにあたって、タスクkは内部メモリ110に必要なメモリ領域を動的に確保する。まずタスクkはOSにメモリ領域を要求する。OSはタスクkに使用可能なメモリ領域、たとえばアドレスZから始まるメモリ領域を割り当てる。タスクkはこの領域を、タスクkのみが排他的にアクセスできる領域として使用するために、領域の秘密保護属性を設定す

10

20

30

40

50

る。秘密保護属性は、タスクkがアドレス領域を指定し、明示的に秘密保護属性設定命令を発行することによって設定される。秘密保護属性の設定は、このタスクのみが行うことができ、たとえOSであっても他のタスクの保護属性を設定することはできない。

【0048】なお、通常、OSは他のタスクとのメモリ資源をめぐる競合を避けるため、メモリ管理ユニットMMU（不図示）に当該領域の資源保護属性を設定して、他のタスクがアクセスできないようにする。この設定はあくまでOSによる資源保護を目的としたものであり、悪意のあるOSの場合、タスクの秘密を保護する役割をなさないことに注意が必要である。

【0049】タスクkから保護属性設定命令が発行されると、処理はプライバシマネージャ301に移行する。プライバシマネージャ301は、MPAテーブル501の指定された領域に対応するエントリに、命令の発行元であるタスクkのIDを書き込み、この領域を他のタスクによるアクセスから保護する。MPAテーブル501の形式は、たとえば図4に示すものである。PAMテーブル502は、各メモリページに対応するエントリ501-1～501-nを有する。エントリは2つのフィールドを有し、フィールドPには、対応するメモリページの秘密保護の有無を示す秘密保護ビットを格納する。たとえば、秘密保護ビットの値が1のときは、秘密保護ありを意味する。残りのフィールドにはタスクIDが格納される。なお、上述したようにプロセッサが資源管理のためのMMU（不図示）を持つ場合には、プライバシマネージャ301が、保護属性設定命令の発行元タスクが対象領域のアクセス権を持っていることを確認する構成とすることが望ましい。

【0050】保護タスクによるメモリアクセス

タスクが保護属性の設定されたアドレスをアクセスする際には、アクセスしようとしている対象領域が保護領域であることを指定してアクセスする。この保護領域の指定は、プロセッサコア102にあるMTR（memory tap e range）レジスタ401によっておこなわれる。

【0051】図5は、MTRレジスタ401の形式を示す。MTRレジスタ401は2ワードからなるレジスタであり、第1のワード401aに対象領域の先頭アドレスを、第2のワード401bにサイズを書き込み、さらに、フィールド402に秘密保護ビット1を設定する。これにより、指定されたアドレス領域にタスクが要求する秘密保護属性（RPA：requested protection attribute）が設定される。要求秘密保護ビットに0を書き込むと、それまで設定されていた秘密保護属性が無効となる。

【0052】あるメモリアクセスが保護になるか、非保護になるかは、MTRレジスタ401によるアドレス範囲の指定と、アクセス対象のアドレスの組み合わせにより決定される。アクセス対象アドレスがMTRレジスタ

401の指定範囲内にあれば、それは保護対象となる。タスクが利用できるMTRレジスタは複数個あってもよい。

【0053】図6は、メモリアクセスの手順を示すフローチャートである。メモリアクセスの手順を説明する。タスクを実行するために、コア102がデータキャッシュ105に読み出し要求を出す時、現在実行中のタスクIDとタスクにより要求される要求秘密保護ビット（RPA）とが、アドレスとともに出力される（S601）。一方、対象領域のアクセス時には、内部メモリ110のユーザエリア111から内容がデータキャッシュ105に読み込まれる。このとき、MPAテーブル501からメモリアドレスに対応するエントリが読み出され、エントリに記述されたタスクIDとメモリ秘密保護ビット（MPA）が、対応するキャッシュラインのタグに書き込まれる（S602）。メモリ秘密保護ビット（MPA）と要求秘密保護ビット（RPA）との取得順序は不問であり、同時に行われてもよい。

【0054】メモリアクセス時にコア102から出力される要求秘密保護属性（RPA）は、あるタスクから見てアクセス対象のメモリに設定されているべきメモリ保護属性である。したがって、この要求秘密保護属性（RPA）と、アクセス対象である内部メモリ領域の秘密保護属性（MPA）とは一致していなければならない。

【0055】そこで、ステップS603で、メモリ保護モジュール106において、キャッシュタグに書き込まれたメモリ秘密保護ビット（MPA）と、コア102から出力された要求秘密保護ビット（RPA）が比較される。比較結果が一致する場合（S603でYES）はステップS604に進み、今度はキャッシュタグに書き込まれたタスクIDと、コア102から出力されたタスクIDが一致するかどうかを判断する。S603とS604の判断順序は問わず、同時であってもよい。

【0056】秘密保護ビットとタスクIDの双方が一致した場合にのみステップS605へ進み、実行しようとするタスクにメモリページのアクセス権があるかどうかを判断する。これは、OSによる資源保護の関係で、メモリ管理ユニットMMUにおいてタスクによる対象メモリ領域のアクセスが禁止されている場合があるからである。アクセス権が認められていれば（S605でYES）、対象のメモリ領域にアクセスが許可され、ステップS606でメモリ操作が実行される。アクセスが禁止されている場合は（S606でNO）、ステップS608で資源保護例外が発生する。

【0057】一方、ステップS603で、キャッシュタグに書き込まれたメモリ秘密保護ビットMPAと、コア102から出力されるタスク要求による秘密保護ビットRPAとが不一致の場合（S603でNO）、あるいは、ステップS604でタスクIDが不一致の場合は、ステップS607でプロセッサコア102に対して秘密

保護例外の発生を通知する。この結果、このタスクの実行は中断され、再開は不可能となる。

【0058】たとえば、タスクkが対象領域へアクセスしようとするときに、MTRレジスタ401に正しい保護領域が設定されておらず、秘密保護されている対象領域に対して秘密保護なしとしてアクセスした場合は、たとえタスクIDが一致していても、メモリ保護モジュール106はアクセス違反と判断する。また、コア102から出力される要求保護ビットRPAが、秘密保護指定してメモリアクセスしようとしているときに、キャッシュタグに書き込まれたメモリ保護ビットMPAが秘密保護なしの領域を対象としている場合も同様に、アクセスは許可されない。

【0059】このような秘密保護例外の判断基準と、資源保護例外の判定基準は、互いの判定結果や順序に依存せず独立である。

【0060】なお、OSとのデータ交換で、非保護メモリをアクセスする場合には、それはMTRレジスタ401に指定した領域外であり、かつMPAテーブル501のエントリの秘密保護ビットは0のはずなので、アクセスは当然成功する。

【0061】もしタスクkのために割り当てられた内部メモリのメモリ領域を、他のタスクがアクセスした場合、コア102から送られるタスクIDと、キャッシュタグのタスクIDとが一致しないため、当然秘密保護例外が発生し、違反アクセスをしたタスクの実行が中断される。

【0062】なお、プログラム領域における保護プログラム領域の保護方法には触れていなかったが、上述したタスク処理と同様に、タスク生成命令によるプログラムのロード時に、対応する領域のMPAテーブル501に秘密保護属性を設定することによって実現される。プログラム領域のメモリアクセスの場合には、要求される秘密保護属性RPAはMTRレジスタ401によって定義されるのではなく、実行中のタスクIDがそのまま使われる。

【0063】タスクの切替

メモリ操作が適正におこなわれタスクkが実行されているときに、割り込みが発生すると、その時点でのワーキングレジスタ114の情報は、図3に示すように、コンテキストテーブル116のタスクID=kのエントリに一時的に退避され、OSに制御が移される。このコンテキストテーブル116の内容はプライバシーネージャ301のみが知ることができ、OSや他のタスクがその内容を見ることはできない。同じくコア102のMTRレジスタ401も、タスクkが要求する要求保護属性(RPA)を指定するため、タスクkのレジスタ情報として、他のタスクによるアクセスから保護される。

【0064】OSによるメモリの解放

OSは、秘密保護属性が設定されたメモリを、強制的に

解放して他の目的に使うことができる。OSは、解放対象のメモリ領域を指定して秘密保護強制解放命令を発行する。命令の発行により、プライバシーネージャ301は対象領域のMPAテーブル501を保護無しに書き換えると同時に、対応するユーザメモリ領域111を所定の値に初期化して、格納されていた情報を消去する。

【0065】秘密保護属性が保護無しに書き換えられたことによって、OSはこのメモリ領域を他のタスクに割り当てる、あるいはOS自身で使うなど、別の用途に使うことができる。

【0066】このメモリ解放機能により、秘密の保護と、OSによる適切なメモリ資源管理が両立する。

【0067】タスクの終了

OSは、秘密保護されたタスクを終了する時、このタスクIDを指定して、タスク終了命令を発行する。タスク終了命令の発行により、プライバシーネージャ301は、タスクの秘密保護属性が設定されたMPAテーブル501の対応するエントリを、保護無しに値に書き換える。同時に、そのメモリ領域をメモリ解放時と同様に初期化する。このタスク終了機能により、タスクが保持する秘密を漏洩することなく、タスクIDを再利用することができる。

【0068】メモリアクセスの統合性保証効果

上記したマイクロプロセッサ100の構成により、メモリアクセスの保護が一貫して保証できることを説明する。

【0069】タスクT₀が、あるアドレスA₀のメモリページM₀に対して、M₀の初期状態からOp₁～Op_nまでn回のメモリ操作をおこなうとする。初期状態は不定であってよいが、Op₁～Op_nの操作を確実にこなって、最終状態の結果を得る必要がある。その過程で別のタスクによるメモリ操作が加わったり、Op₁～Op_nの操作順序が変更されたり、省略されてもならない。一連の操作の途中で一度ページが初期化されても、結果は期待と異なるものになってしまう。

【0070】第1実施形態の方法では、最初にタスクT₀がメモリページM₀を確保して秘密保護属性を設定する命令を発行する。これにより、メモリページM₀に対応するMPAテーブル501のエントリに、秘密保護ビット“1”とタスクIDを設定する。その後、MTRレジスタ401にM₀のアドレスを設定して、Op₁～Op_nの操作を行う。プログラム自身は暗号化されており、攻撃者が順序変更などを行うことは困難であり、タスク自身によるメモリページの秘密保護属性設定、MTTレジスタ301の設定、Op₁～Op_nの操作の順序が正しいことは保証される。

【0071】次に、メモリページM₀は実アドレスによって参照されるため、アドレスとして参照されるページは、常にただひとつだけであることが保証される。

【0072】したがって、Op₁～Op_nのアクセス

10

20

30

40

50

において、秘密保護例外が発生せずアクセスが成功したということは、確かに対象メモリページ M_0 に対して $Op_1 \sim Op_n$ の操作が行われたということになる。

【0073】また、あるメモリページに秘密保護属性を設定できるのは、タスク自身であり、このタスクは、 $Op_1 \sim Op_n$ の操作の前に、1回だけしか秘密保護属性の設定を行っていない。メモリページへの $Op_1 \sim Op_n$ のアクセスは、MTRレジスタ401において秘密保護ビット1を指定して行なわれている。仮にこの間にOSがメモリページの解放を行っていたとすると、 $Op_1 \sim Op_n$ においてMTRレジスタ401で指定された保護属性(RPA)と、メモリページの保護属性(MPA)の不一致が生じ、秘密保護例外が発生してタスクの実行はそこで中止されていたはずである。

【0074】よって、 $Op_1 \sim Op_n$ のアクセスにおいて、秘密保護例外が発生せずアクセスが成功したということは、これらの操作の間に一度もOSによるメモリの解放が行われていないといえる。

【0075】このように、第1実施形態のマイクロプロセッサの構成により、上述したメモリ操作の統合性、一貫性を保証することができる。そして、このメモリ操作の統合性は、OSによる強制的メモリ解放(資源管理)と、タスクによる非保護メモリページのアクセスと両立している。

【0076】このような資源管理と秘密保護の両立は、タスクのみがメモリの秘密保護属性を設定できることと、メモリアクセス時に、メモリページに設けられたメモリ保護属性MPAと、タスクが保持する要求保護属性RPAを比較して属性が一致した場合のみアクセスを許可することを組み合わせた結果、実現されるものである。

【0077】もっとも、MPAとRPAが一致した場合のみメモリアクセスを許可するという構成は、保護メモリと非保護メモリのアクセスが混在する場合において、タスクによる明示の秘密保護属性設定がなくても、単独でメモリ操作の安全性を高める効果がある。

【0078】<第2実施形態>第1実施形態では、内部メモリの秘密保護属性の設定は、タスクが発行する明示的な命令によりおこなわれていた。第2実施形態では、メモリの秘密保護属性の設定は、メモリアクセスによって暗黙に行なわれる。すなわち、メモリページが秘密保護されていない状態(すなわちメモリ保護属性MPAが0)のときに、タスクが秘密保護を要求する状態(すなわち要求保護属性RPAの値が1)でメモリにアクセスすると、自動的に対応するMPAエントリのフィールドに、メモリ保護ビット1が設定される。同時にメモリ内容が乱数により初期化される。この構成により、メモリ内容の安全性はさらに向上する。

【0079】メモリアクセス時に、対象メモリ領域に秘密保護属性定が黙示的に設定されること以外は、第1実施形態と同様である。以下、メモリアクセス時の黙示の

秘密保護属性設定について、詳細に説明する。

【0080】図7は、黙示の秘密保護属性設定を含むメモリアクセスの手順を示すフローチャートである。あるメモリ領域にアクセスするにあたって、それが保護領域としてアクセスされるか否かは、第1実施形態と同様に、MTRレジスタ401のアドレス範囲指定と、対象アドレスの組み合わせにより決定される。

【0081】まずステップS701で、要求秘密保護属性(RPA)を構成する秘密保護ビットとタスクIDが、対象アドレスとともにコア102から出力される。一方、ステップS702で、キャッシュタグに格納されている対応するメモリページのMPA保護ビットとタスクIDがメモリ保護モジュール106で取得される。

【0082】ステップS703でRPAとMPAの秘密保護ビットの一致が判定される。一致した場合、ステップS704でタスクIDの一致が判定され、あとは第1実施形態と同様に、S705、S706、S711で資源保護判断が処理される。

【0083】一致しない場合は(S703でNO)、ステップS707に進み、RPAの秘密保護ビットが1であるか否かを判断する。RPAの秘密保護ビットが1ならば(S707でYES)、ステップS708で処理はプライバシマネージャ301に移り、アクセス対象メモリページのMPAエントリに、秘密保護ビットの値1と、アクセスしたタスクのタスクIDが設定される。さらに、ステップS709で、乱数によってそのページの内容がすべて初期化される。その後、ステップS704へ進んで、タスクIDの一致が判断され、以後の処理は第1実施形態と同様である。

【0084】ステップS707でRPAの秘密保護ビットが1でないとき(すなわち0のとき)は、そのアクセスは秘密保護違反として扱われ、ステップS710で秘密保護例外が発生する。

【0085】第2実施形態の構成により、OSによる意図なメモリ改変から、タスクのメモリアクセスが一貫して保護されることを補足説明する。

【0086】タスク T_0 が、あるアドレス A_0 のメモリページ M_0 に対して、 M_0 の初期状態から $Op_1 \sim Op_n$ まで n 回のメモリ操作をおこなうとする。初期状態は不定であってよいが、 $Op_1 \sim Op_n$ の操作を確実にこなって、最終状態の結果を得る必要がある。その過程で別のタスクによるメモリ操作が加わったり、 $Op_1 \sim Op_n$ の操作順序が変更されたり、省略されてもならない。一連の操作の途中で一度ページが初期化されても、結果は期待と異なるものになってしまう。

【0087】第2実施形態の方法では、タスク T_0 はMTRレジスタ401にメモリページ M_0 のアドレスを設定して、メモリページ M_0 に対して Op_1 のメモリ操作を発行する。その際、メモリページ M_0 にメモリ秘密保護属性が設定される。初期状態では、メモリページ M_0

には保護属性がまだ付与されていないはずである。タスクT₀はその後、Op₁～Op_nを行う。

【0088】まずプログラム自身は暗号化されており、攻撃者が順序変更などを行うことは困難である。したがって、メモリページの秘密保護属性設定、MTRレジスタ401の設定、Op₁～Op_nの操作の順序が正しいことは保証される。

【0089】次に、メモリページは実アドレスによって参照されるため、あるアドレスで参照されるページは常にただひとつだけである。したがって、Op₁～Op_n

のアクセスにおいて、秘密保護例外が発生せずアクセスが成功したということは、確かに対象メモリページM₀に対して、Op₁～Op_nの操作が行われたということになる。

【0090】また、あるメモリページにそのタスクの秘密保護属性を設定できるのは当該タスクだけであり、このタスクによって秘密保護属性が設定されるのは、Op₁～Op_nの操作の際に限られる。ただし、その際にはメモリページの内容は乱数によって初期化されてしまう。

【0091】したがって、この間にOSが一旦メモリページの解放を行い、M₀の内容を意図したように書き換えたとしても、その場合、保護属性がクリアされてしまう。この結果、内容は必ず乱数により初期化されてしまい、意図的な改変は成功しない。もちろん、タスクの正常な動作は妨害されるが、その結果が攻撃者の意図した結果を生む可能性は一般に極めて低いといえる。

【0092】第2実施形態の構成により、タスクの秘密メモリを意図的な改変から保護することができる。このような保護は、OSによる強制的メモリ解放およびタスクによる非保護メモリページのアクセスと両立している。

【0093】＜第3実施形態＞第1および第2の実施形態では、実アドレス方式のマイクロプロセッサを対象として、タスクの秘密の安全性とOSによるメモリ管理が両立し、同時に、タスクとOSの通信が確保され、メモリ操作の一貫性が保証される秘密保護方式を説明した。

【0094】第3実施形態は、仮想記憶機構を持つ内部メモリ型プロセッサに関する。特に、仮想記憶機構をもつプロセッサにおいて問題となるメモリページすりかえ攻撃の防止を説明する。

【0095】仮想記憶機構を持つシステムでは、OSは物理メモリの任意のページを任意の仮想アドレスに割り当てることができる。また、この割当はタスクに応じて異なる割当を使うことができる。割当の記述には、ページテーブルと呼ばれる内部メインメモリ上の表が使われる。ページテーブルについては、後述する。

【0096】メモリアクセスにあたって、仮想アドレスを高速に物理アドレスに変換するために、ページテーブルの必要な部分だけを特殊なキャッシュメモリに読み込

んでおくことが多い。この一時メモリはTLB (translation load-aside buffer) と呼ばれる。

【0097】ページテーブルのTLBへの読み込みは、プロセッサハードウェアによっておこなわれるものと、ソフトウェアによるものの2種類がある。前者の例にはIntel社のPentiumがあり、後者の例にはMIPS社のR3000がある。第3実施形態でのTLBへの読み込みは、どちらの場合にも適用可能である。

【0098】図8は、第3実施形態のマイクロプロセッサにおけるTLB構成を示す。第3実施形態での秘密保護属性の設定は、第1実施形態と同じくタスクによる明示の設定とする。したがって、TLB構成以外の部分については、図1に示すものと同様である。

【0099】まず、内部メモリ110のユーザエリア111に、ページテーブル111-1が設定される。また、MPAテーブル501のエントリ(図4参照)に、タスクIDに加えて当該メモリページに指定された仮想アドレスを格納するフィールドを含む。

【0100】次に、命令キャッシュ104はP-TLB(104-1)とR-TLB(104-2)を有し、データキャッシュ105も同様に、P-TLB(105-1)とR-TLB(105-2)を有する。R-TLBはアドレス変換のためのTLBである。一方、P-TLBはメモリ秘密保護属性を格納するものであり、R-TLBを補完してタスクの秘密保護をより安全にする。R-TLBとP-TLBのエントリは一対一対応に存在し、R-TLBへ書き込みを行うことによって、後述する処理がP-TLBに対して自動的に行われる。換言すると、P-TLBの更新は、R-TLBの書き込みを契機として間接的に行なわれ、プライバシマネージャ301以外がP-TLBを明示的に操作することはできない。

【0101】第3実施形態においては、タスクの秘密保護属性の要素として属性指定時の仮想アドレスが保持される。上述したように、第1実施形態と同様に、タスクによる秘密保護属性の設定命令の発行時に、仮想アドレスの設定が行なわれるものとする。仮想アドレスには、タスクが指定したアドレスが使われる。ここでは物理アドレスP₀のメモリページM₀が仮想アドレスV₀にマップされているものとする。この時、プライバシマネージャ301が保護属性を書き込むMPAテーブル501のエントリとして、メモリページM₀の物理アドレスP₀に対応するエントリが選択される。選択されたMPAエントリには、タスクIDと、メモリページM₀の仮想アドレスV₀が書き込まれる。物理ページに割り当てられる仮想アドレスはタスクによって異なる場合があるが、第3実施例ではタスク間の保護メモリ共有を考えていないため、当該メモリに保護属性が設定されている以上、他タスクのアクセスはありえないので問題はない。

【0102】図9は、第3実施形態のマイクロプロセッ

サにおけるメモリアクセスの手順を説明する。

【0103】ステップS801において、あるメモリページM₀がアクセスされる時、データの読み込みに先立って、アクセス対象の仮想アドレスV₀に対応するエントリが取得される。ステップS802で、取得したエントリを、ページテーブル111-1からデータキャッシュ105のR-TLBにロードして、仮想アドレスV₀から物理アドレスP₀への変換の準備が行われる。R-TLBへのロードは、ハードウェアまたはソフトウェア(OS)によって行われる。

【0104】ロードが行われると、ステップS803で自動的に、変換先の物理アドレスP₀に対応するMPAエントリがプライバシマネージャ301によって読み出される。ステップS804で、MPAテーブル501から読み出された設定時の仮想アドレスV₀が、変換元であるR-TLMの仮想アドレスと照合され、一致するかどうか判断される。

【0105】一致すれば(S804でYES)、TLBロードは成功する。一致しなければ(S804でNO) TLBロードは失敗し、ステップS806で秘密保護例外が発生する。

【0106】TLBのロードが成功すると、対応するデータキャッシュのP-TLB(105-1)に、MPAエントリのタスクIDと、秘密保護の有無が書き込まれる。さらに、R-TLB(105-2)を参照したアドレス変換の結果読み込まれるキャッシュラインのタグに、対応するP-TLB(105-1)のエントリに書かれたタスクIDと秘密保護の有無が書き込まれる。その後の処理は第1実施形態の処理と同様である。

【0107】第1実施形態においては、物理メモリ対応に保持されるメモリ保護属性(MPA)とタスクが要求する要求保護属性(RPA)には、それぞれ秘密保護ビットとタスクIDが含まれ、その比較はコアからのメモリアクセス要求時に行なわれていた。

【0108】第3実施形態においては、メモリ保護属性(MPA)と要求保護属性(RPA)の比較は、ページ単位に行なわれるTLBのロード時と、ワード単位に行なわれるメモリアクセス時に分散して行われる。TLBのロード時には、タスクが要求する仮想アドレスV₀と、MPAテーブル501に保持された保護属性設定時の仮想アドレスが比較されている。メモリアクセス時には、第1実施形態と同様に、秘密保護ビットとタスクIDが比較されている。

【0109】第3実施形態でも基本的には、タスクの実行の結果生成された要求保護属性RPAと、対象領域であるメモリの保護属性を記述するメモリ保護属性MPAとが比較されているが、比較が属性の項目毎に、時間的に分散して、異なる粒度でなされている点が異なる。

【0110】第3実施形態の構成は、以下のような攻撃に対して、メモリ操作の統合性を保証できる点で有効で

ある。

【0111】あるタスクについて、保護属性が設定されたメモリページM₀、M₁を、それぞれ仮想アドレスV₀、V₁に配置し、その後処理Op₁~Op_nを実施する場合を考える。Op₁~Op_nの処理はあらかじめV₀とV₁のどちらで行われるか決まっている。

【0112】ところが、仮想記憶機構を使えば、Op₁~Op_nの操作の任意の時点で物理メモリページM₀、M₁の仮想アドレス上の配置を入れ換えることができる。例えば、メモリページM₁に対する操作がメモリクリアであれば、操作対象のメモリページの仮想アドレスを入れ換えることで、メモリページM₀をクリアすることができてしまう。この場合、処理はプログラム作成者の意図とは異なったものとなるが、第1実施形態のような実アドレスのみを前提とした構成では、このようなすり替え攻撃を防ぐことはできない。

【0113】第3実施形態では、メモリ秘密保護属性設定時に参照した仮想アドレスを、物理メモリページの秘密保護属性定義として保存しておき、アドレス変換の実行時に変換対象の仮想アドレスと、保存されたアドレスとを比較する。そして不一致が生じた場合にアクセスを禁止することにより、上述したメモリページ再配置攻撃を防止している。第3実施形態では、TLBのロード時に比較処理することで処理の効率化をはかっているが、プロセッサコアによるメモリ参照の都度、比較を行ってもよい。この場合、効率は多少低下するものの、同様の効果が得られる。

【0114】なお上記の例では、仮想メモリを用いた場合のメモリアクセスの統合性保証を、タスクが明示的にメモリ秘密保護属性設定命令を発行する構成と組み合わせで説明した。しかし、仮想アドレスの比較処理そのものは、単独で使用しても仮想記憶を利用したメモリ交換攻撃の防止に有効であることを付け加えておく。

【0115】また、秘密保護属性命令の発行時には、対応する物理アドレスの変換を定義したP-TLBをいったんフラッシュしておくことが望ましい。秘密保護属性命令の発行前に読み込まれたTLBエントリが残存する場合、不都合が生じるからである。たとえば、非保護状態のTLBエントリを参照した結果読み込まれたキャッシュラインをタスクが参照すると、期待に反して秘密保護属性が設定されておらず、秘密保護例外が発生して処理が中止されてしまう。そこで、秘密保護属性命令発行時には、P-TLBをフラッシュすることが望ましい。

【0116】＜第4実施形態＞第1～第3実施形態では、内部メモリ型耐タンパプロセッサにおいて、タスクによる排他的なメモリアクセスを実現する構成を説明した。

【0117】第4実施形態では、秘密を共有する複数のタスクが、あるメモリ領域を共有でき、かつ秘密を保持

しないタスクには排他的なアクセスを可能にする構成例を説明する。

【0118】第4実施形態においても、基本構成は図1に示す第1実施形態の構成と同様である。したがって、以下では変更点を中心に説明することとする。

【0119】まず、タスクが内部メモリの共有領域にアクセスするための秘密保護情報はMTRレジスタ401に保持される。第1および第2実施形態では、MTRレジスタ401にはアドレス領域のみが保持されていたが、第4実施形態では、アドレス領域に加えて、共有の

ための秘密値が保持される。秘密値はタスクが秘密保護属性設定時に指定する。そして、この秘密値はメモリ参照時に要求秘密保護属性(RPA)として使用される。なお、秘密値の長さは総当たり攻撃を防止するため、128bit以上であることが望ましい。

【0120】また、この秘密値は、タスクによりメモリページに秘密保護属性が設定される時にも指定され、MPAテーブル501の対応するエントリに格納される。したがって、第4実施形態では、MPAテーブル501のエントリに格納されるのは、保護の有無、タスクID、仮想アドレス、秘密値の4項目となる。

【0121】プログラムのロード時に、プログラム領域のメモリ保護属性(MPA)が設定される場合、MPAエントリの秘密値には、プログラムを暗号化していたプログラム鍵の値が書き込まれる。

【0122】図10は、第4実施形態のマイクロプロセッサにおけるメモリアクセスの手順を示す。

【0123】アクセスの可否は、すでに説明した第1～第3実施形態と同様、プロセッサコア102から出力される要求秘密保護属性(RPA)と、キャッシュタグに格納されたメモリ秘密保護属性(MPA)の一致によって判定される。

【0124】すなわち、ステップS901で、コアからは保護の有無を示す秘密保護ビット、タスクID、仮想アドレス、秘密値が得られ、ステップS902で、キャッシュタグに書き込まれたメモリ保護ビット、タスクID、仮想アドレス、秘密値が得られる。コアからの出力される4項目のうち、要求秘密保護属性(RPA)として使用されるのは、タスクIDを除く3項目である。これは、タスク間でメモリが共有されるため、判定基準をタスクIDに依存せず、秘密値の合致におきかえたためである。

【0125】ステップS903で、メモリ保護モジュール106において、コアから出力されたRPAの秘密保護ビットと、キャッシュタグに書き込まれたMPAの秘密保護ビットが比較される。不一致の場合は、第1～第3実施形態と同様に、ステップS908で秘密保護例外が発生する。

【0126】ステップS903で、両者の秘密保護ビットが一致する場合は、ステップS904、S905で、

両者(RPAとMPA)の仮想アドレスと秘密値とが、それぞれ比較される。一致しない場合は、ステップS908で秘密保護例外が発生する。

【0127】秘密保護の面で秘密保護ビット、仮想アドレス、秘密値のすべてが一致したならば(S903～S905ですべてYES)、ステップS906に進み、資源保護の面でタスクIDが一致するかどうかを判断する。不一致の場合はステップS909で資源保護例外が発生する。一致する場合は、ステップS907でメモリ操作が実行される。

【0128】第4実施形態のマイクロプロセッサの具体的な利用態様としては、たとえば複数のプログラムにあらかじめ共通の秘密Xが埋め込まれており、あるプログラムから生成されたタスクAが共有メモリ領域を確保して秘密値Xを設定しておき、別のタスクBがMTRレジスタ401に秘密値Xを設定してアクセスする例が挙げられる。秘密値Xは、あらかじめ定められたもの以外に、Diffy-Hellman法などのよく知られた鍵共有方式によって共有されたものでもよい。

【0129】図10のフローチャートで説明したように、秘密値Xを知らないタスクによるこのメモリ共有領域へのアクセスは秘密保護例外となり、当然失敗する。したがって、秘密値Xを知るタスクだけの排他的なメモリ共有が実現できる。

【0130】第4実施形態では、メモリ領域の排他的な(安全な)共有を、タスクによるメモリ秘密保護属性の設定命令と組み合わせて説明した記述した。しかし、秘密値をMPA、RPAに利用してメモリ領域の共有を安全に行う方法は、タスクが発行した秘密保護属性の設定命令の存在に依存するわけではなく、単独でタスク間の安全なメモリ共有を可能にすることを付記しておく。

【0131】

【発明の効果】以上説明したように、内部メモリ型のマイクロプロセッサにおいて、マルチタスク環境下において、OSによる内部メモリの資源管理と秘密保護とを両立させ、OSとタスクとの通信を確保しつつ、複数プログラムを意図的改変から保護することができる。

【図面の簡単な説明】

【図1】本発明の第1実施形態に係る内部メモリ型耐タンパプロセッサの概略構成図である。

【図2】図1の耐タンパプロセッサにおける内部メモリマップの図である。

【図3】図1の耐タンパプロセッサにおけるタスク開始とメモリ確保を説明するための図である。

【図4】図1の耐タンパプロセッサにおけるMPAテーブルの構成例を示す図である。

【図5】図1の耐タンパプロセッサにおけるMTRレジスタの構成例を示す図である。

【図6】図1の耐タンパプロセッサのメモリアクセス手順を示すフローチャートである。

【図 7】本発明の第 2 実施形態に係るメモリアクセス手順を示すフローチャートである。

【図 8】本発明の第 3 実施形態に係る耐タンパプロセッサの秘密保護機構の概略構成図である。

【図 9】第 3 実施形態の耐タンパプロセッサのメモリアクセス手順を示すフローチャートである。

【図 10】本発明の第 4 実施形態の耐タンパプロセッサにおけるメモリアクセス手順を示すフローチャートである。

【図 11】一般的なマルチパーティのアプリケーション保護環境を示す概略図である。

【符号の説明】

100 内部メモリ型耐タンパプロセッサ
101 プロセッサパッケージ
102 プロセッサコア
103 キャッシュモジュール
104 命令キャッシュ
105 データキャッシュ
106 メモリ保護モジュール
107 BIU
108 暗号モジュール
109 ユーザエリア
110 秘匿エリア
111 MPA テーブル
112 ユーザエリア
113 秘匿エリア
114 ワーキングレジスタ
115 MTRR
116 タスクコンテキストテーブル
117 タスクキーテーブル
118 インタラプト(割込)
119 ファームウェア
120 プライバシマネージャ
121 フラッシュメモリ
122 周辺サウスブリッジ
123 バッファメモリ
124 HDD
125 ネットワーク

* 105 データキャッシュ

104-1、105-1 P-TLB

104-2、105-2 R-TLB

106 メモリ保護モジュール (メモリ保護手段)

107 バスインターフェイスユニット

108 暗号モジュール (復号化手段)

109 ファームウェア

110 内部メモリ

111 ユーザエリア

111-1 ページテーブル

112 秘匿エリア

114 ワーキングレジスタ

116 タスクコンテキストテーブル

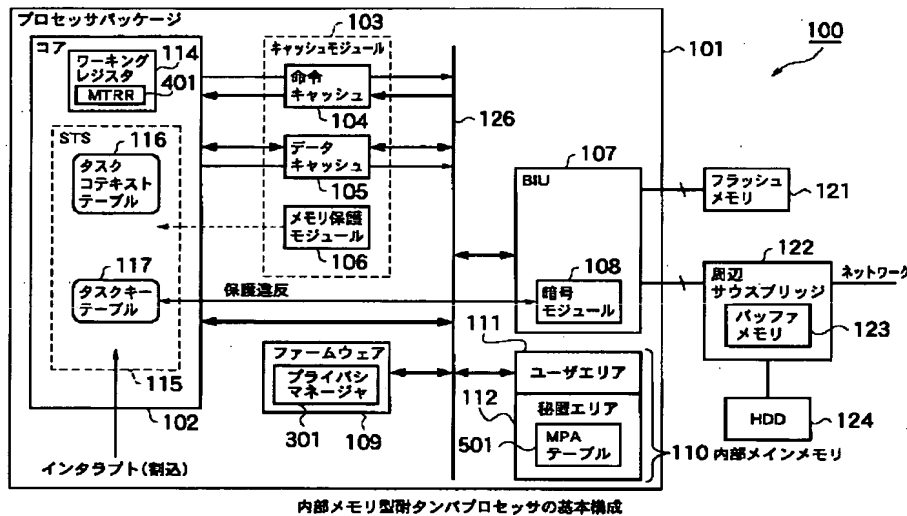
117 タスクキーテーブル

301 プライバシマネージャ (秘密保護管理部)

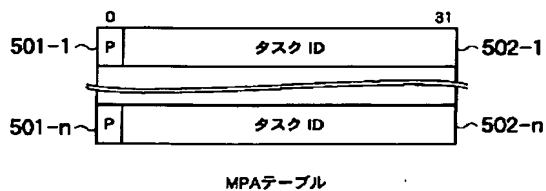
401 MTR レジスタ (要求秘密保護属性保持部)

* 501 MPA テーブル (メモリ秘密保護属性保持部)

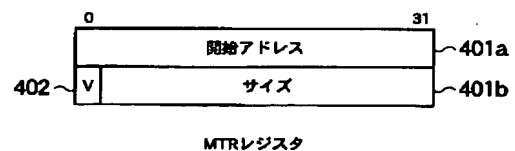
【図 1】



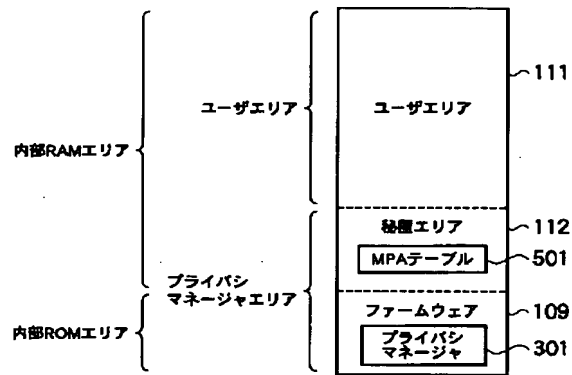
【図 4】



【図 5】

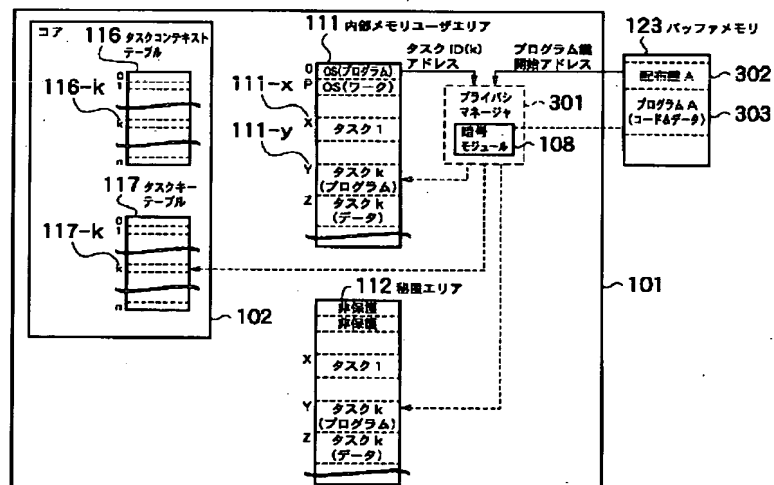


【図2】



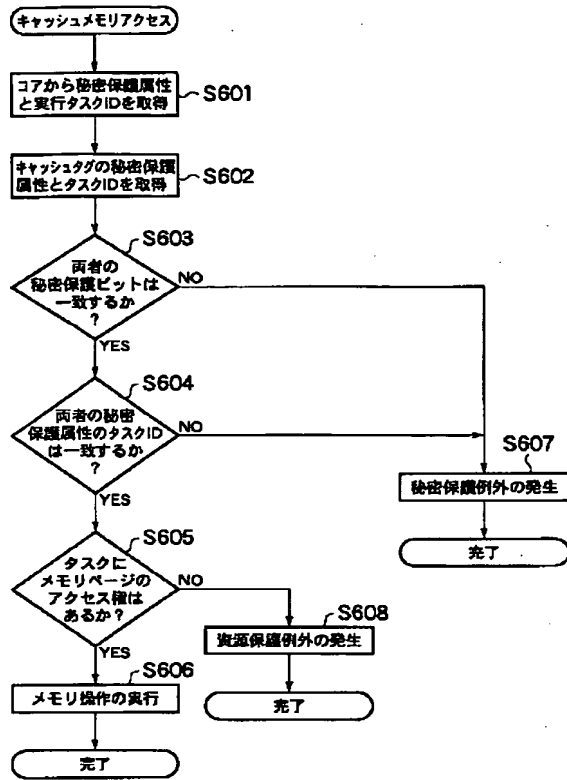
内部メモリマップ

【図3】



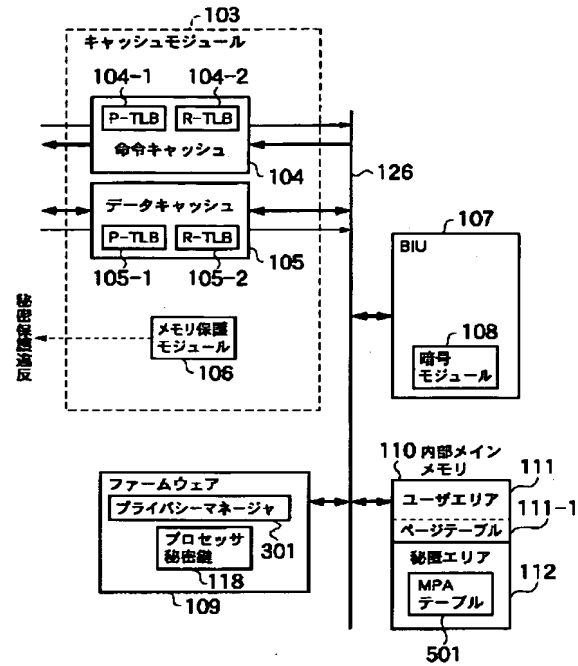
タスク開始およびメモリ確保

【図6】

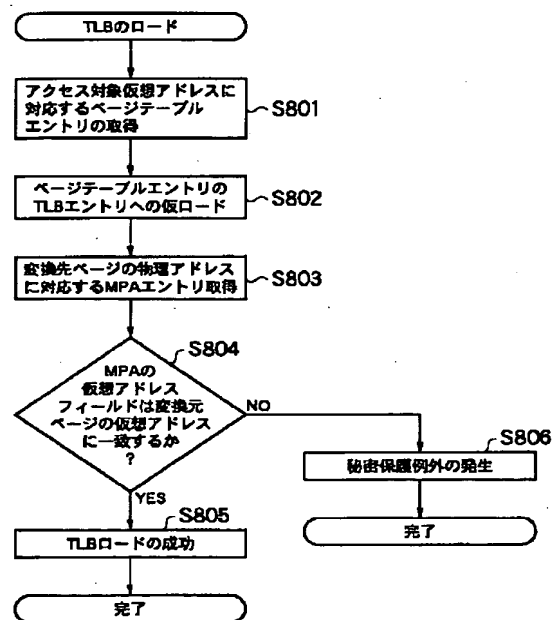


メモリアクセス手順

【図8】

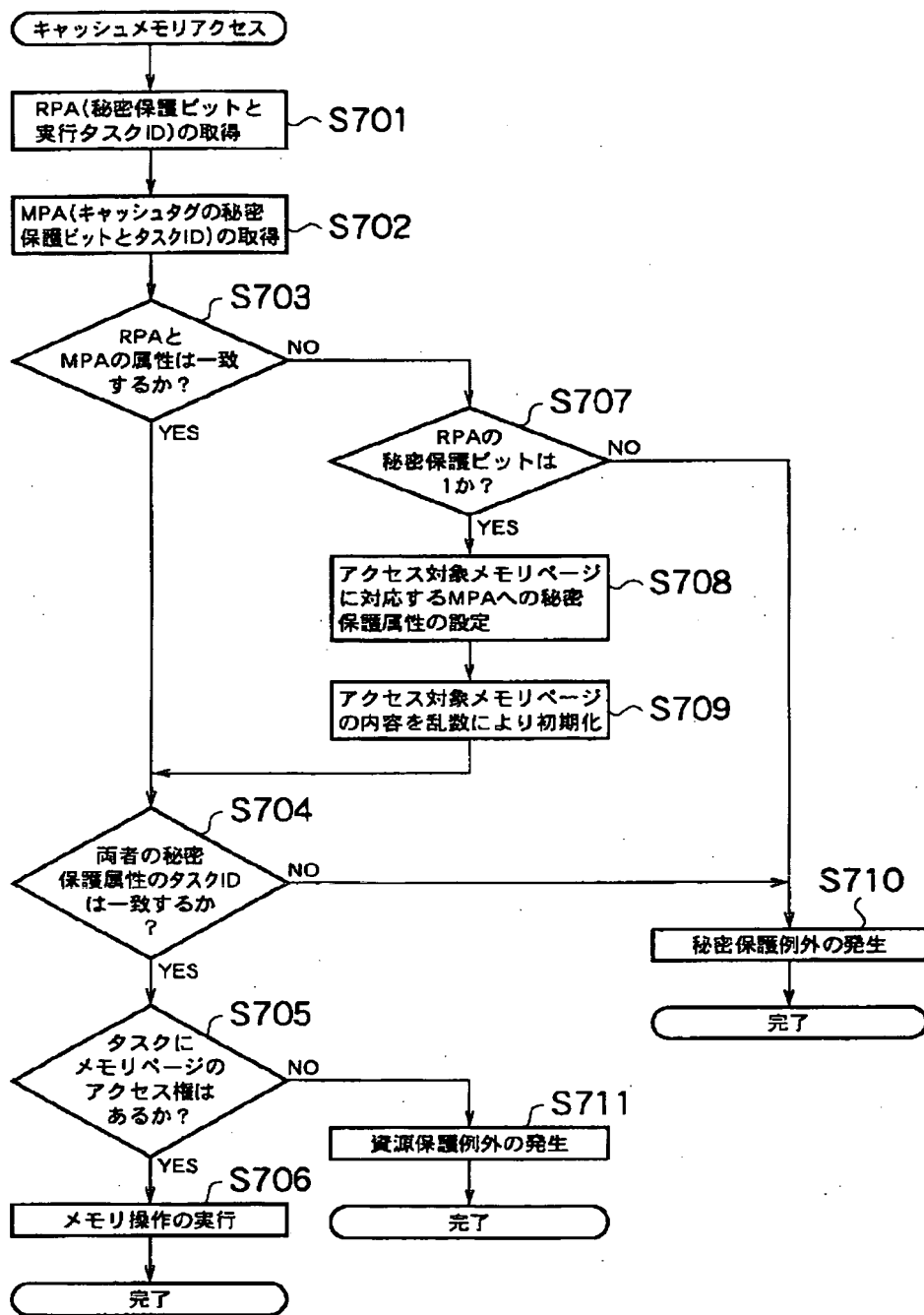


【図9】



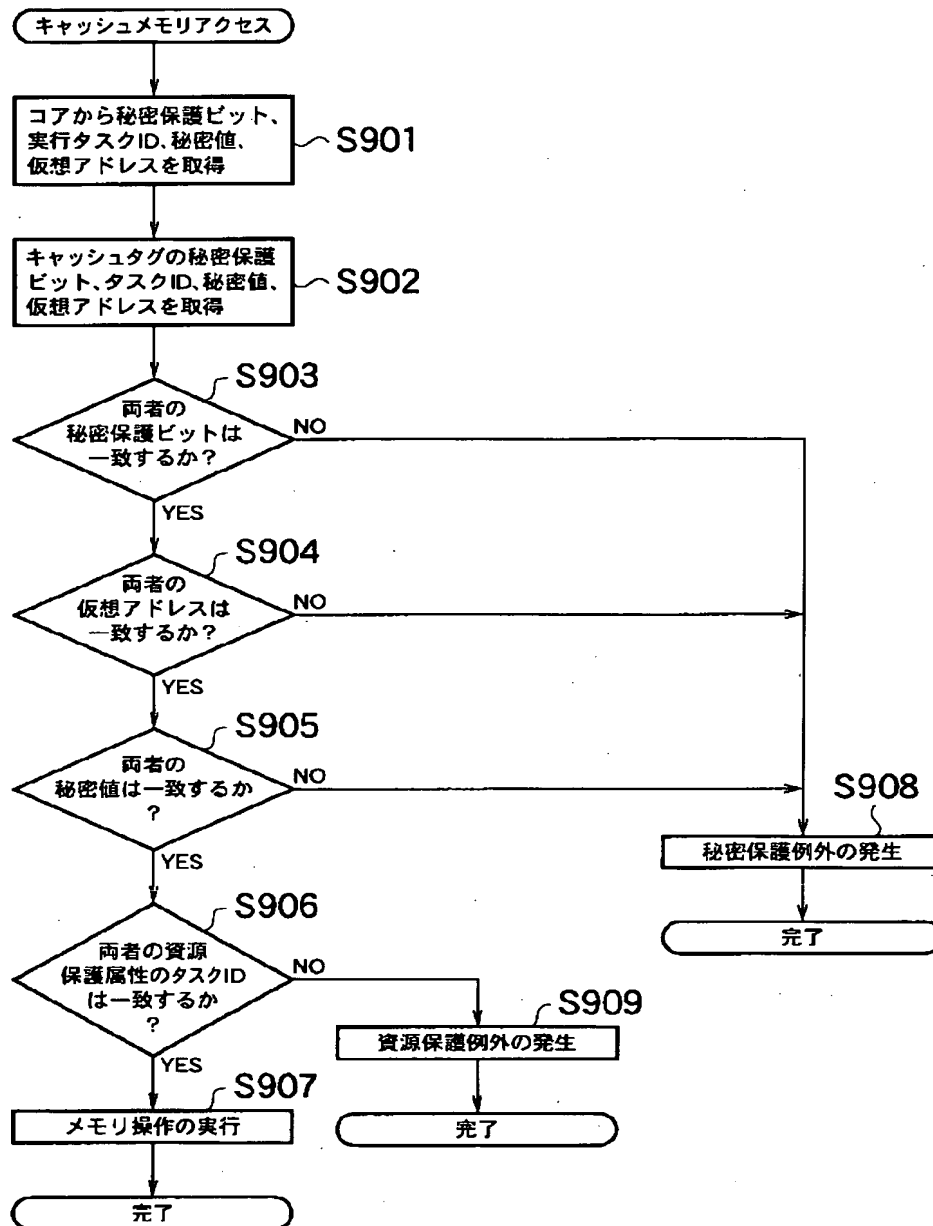
TLB構成

【図7】



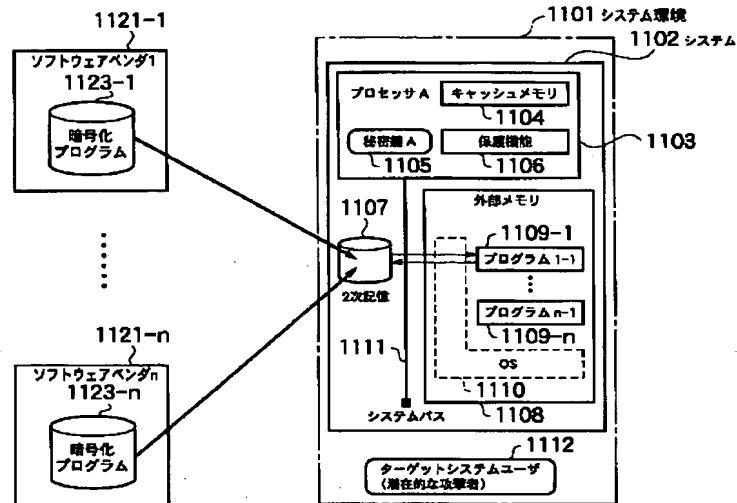
メモリアクセス手順

【図10】



メモリアクセス手順

【図11】



フロントページの続き

(51) Int. Cl.⁷
G 0 6 F 15/78

識別記号
5 1 0

F I
G 0 6 F 9/06

テーマコード (参考)

6 6 0 J

(72) 発明者 白河 健治
神奈川県川崎市幸区小向東芝町1 株式会
社東芝研究開発センター内
(72) 発明者 藤本 謙作
神奈川県川崎市幸区小向東芝町1 株式会
社東芝研究開発センター内

F ターム (参考) 5B005 MM02 MM03 MM52
5B017 AA01 BA02 CA01
5B062 AA07 CC01
5B076 FA01 FC06
5B098 GA04 GD03 GD07 GD14